The Journal of Supercomputing, 34, 181–199, 2005

A General Performance Model of Structured and Unstructured Mesh Particle Transport Computations

MARK M. MATHIS mmathis@lanl.gov
DARREN J. KERBYSON djk@lanl.gov
Performance and Architecture Laboratory (PAL), Los Alamos National Laboratory, CCS-3, P.O. Box 1663, Los Alamos, NM 87545

Abstract. The performance of unstructured mesh applications presents a number of complexities and subtleties that do not arise for dense structured meshes. From a programming point of view, the handling of unstructured meshes has an increased complexity in order to manage the necessary data structures and interactions between mesh-cells. From a performance point of view, there are added difficulties in understanding both the processing time on a single processor and the scaling characteristics when using large-scale parallel systems. In this work we present a general performance model for the calculation of deterministic S_N transport on unstructured meshes that is also applicable to structured meshes. The model captures the key processing characteristics of the calculation and is parametric using both system performance data (latency, bandwidth, processing rate etc.) and application data (mesh size etc.) as input. A single formulation of the model is used to predict the performance of two quite different implementations of the same calculation. It is validated on two clusters (an HP AlphaServer and an Itanium-2 system) showing high prediction accuracy.

Keywords: performance modeling, performance analysis, high performance computing, SN transport, unstructured meshes, parallel processing, large-scale systems

1. Introduction

Performance modeling is an important tool that can be used by a performance analyst to provide insight into the achievable performance of a system and/or an application. Indeed, it is only through knowledge of the workload that a system is to be used for that a meaningful performanc analysis can be made. It has been recognized that performance modeling can be used throughout the life-cycle of a system, or of an application, from first design through to maintenance [6] including procurement and system installation.

Recent work at Los Alamos National Laboratory (LANL) has demonstrated the use of performance modeling in many situations, for instance: in the early design of systems; during the procurement of ASCI purple (expected to be a 100-Tflop system to be installed in 2005); to explore possible optimizations in applications prior to implementation [5]; and to verify the performance of the 20-Tflop ASCI Q system during its installation [8] (which ultimately lead to system optimizations resulting in a factor of 2 performance improvement [15]). Models have also been used to compare large-scale system performance including a comparison of several of the highest peak-rated terascale systems such as the Earth Simulator [7] and BlueGene/L [1] with ASCI Q.

In this work we present the development and use of a model that accurately captures the performance characteristics of deterministic S_N transport on unstructured as well as

structured meshes. This calculation solves the first-order form of the steady-state Boltzmann transport equation. It is representative of applications that use a high percentage of the total cycles across the ASC (Accelerated Strategic Computing) machines.

Unstructured meshes have benefits over structured meshes in terms of representing the geometry being processed, but result with significant extra overhead in terms of performance. Several important performance factors that can reduce the overall calculation efficiency of this type of computation on large-scale parallel systems are analyzed in this paper.

Efforts devoted to the performance analysis of S_N transport date back many years. Research has included the development of analytic performance models as a function of problem mesh and machine size [9]. More detailed performance models have been developed that also include detailed communication [2], and SMP cluster characteristics [3]. Further work has considered different partitioning schemes [10], but all use an underlying structured mesh in their analysis. Particle transport has also been used in empirical performance evaluation studies (e.g., [19]).

The key contribution of this paper is the development of a general analytical performance model of S_N transport computations on unstructured meshes. The model is shown to be applicable across different implementations including an experimental code under development at LANL called Tycho [13], and a production code called UMT2K [18] from Lawrence Livermore National Laboratory (LLNL) as well as being backwardly compatible with the existing performance model of the structured mesh code, Sweep3D [17]. Tycho is written in C++, and UMT2K is written in a combination of Fortran and C. UMT2K was part of the benchmark suite used in the recent procurement of ASCI purple. Other codes are also being developed for S_N transport computations e.g. [16].

The analytical model developed here is shown to have reasonable accuracy in a validation process using a 64-node HP AlphaServer system and a 32-node Itanium-2 cluster. We concentrate on the development and the validation of the model in this work. However, the accuracy of the model is such that it may be used to explore many performance scenarios, for instance to examine the achievable performance that could be obtained on hypothetical future architectures and also to indicate the size of mesh that could be processed in a given time.

This paper represents an extension of work presented in [11]. Our previous work shows that a model for an abstract algorithm can work equally well for different implementations of the same solution. To that end, we have expanded the discussion of the two applications presented here. Furthermore, we demonstrate that the model for unstructured meshes can be reduced to a model for structured meshes. This analysis shows that the model for sweeps on structured meshes is indeed a sub-case of the unstructured mesh model. Finally, we cross-validate one of the codes on a different system. This further illustrates the flexibility of our modeling approach.

The paper is organized as follows. In Section 2, the S_N transport calculation is described and comparisons between its operation on structured and un-structured meshes are made. In Section 3 the key processing characteristics are detailed which are used in the development of the performance model in Section 4. The model is validated in Section 5 on a number of different input unstructured meshes that represent different physical geometries for both Tycho and UMT2K.

2. Overview of S_N transport algorithms

The algorithms employed in deterministic S_N transport (discrete ordinates) computations fall in a class generically named wavefront techniques. In a nutshell they utilize an iterative approach using a method of "sweeping". Each spatial cell in a mesh is processed for each direction in the discrete ordinates set. The wavefronts (or sweeps) are software pipelined in each of the processing directions. Wavefront algorithms exhibit several interesting performance characteristics, related to the pipelined nature of the wavefront dynamics. These include a pipeline delay across processors for a sweep, and a repetition rate of both computation and communication in the direction of the sweep.

In the case of a structured mesh, a high processing efficiency can be achieved as all active processors perform the same amount of work, and communicate the same sized boundary data in each processing step. However, the efficiency when using unstructured meshes is lower due to a possible imbalance of work across processors as the wavefronts progress. The processing flow of the calculation is described below for both structured and unstructured meshes.

2.1. The method of sweeping

Structured meshes. In three-dimensions, each sweep direction can be considered to originate in one of the eight corners ("octants") of the spatial domain. For a typical discrete ordinate order of 6, i.e. S_6, the total number of sweep directions is 48 or 6 per octant. For sweep directions within each octant, the ordering of cell processing on a structured mesh is identical. Figure 1(a) shows the first six steps of two separate sweeps at different angles, Ω, originating from different quadrants for a two-dimensional spatial domain. The edge of the sweep corresponds to a wavefront and is shown as black. Cells on the sweep edge require the grey cells to have been processed in previous steps. The same operation can take place in three dimensions resulting in a wavefront surface. The wavefront propagates across the spatial domain at a constant calculation velocity since the time needed to process a cell is constant. This processing algorithm as developed in [9], uses direct indexing of the spatial mesh as the cell processing order is deterministic for each sweep direction.

Unstructured meshes. An example two-dimensional unstructured mesh consisting of triangles is depicted in Figure 1(b). Two sweep directions are again used to illustrate the processing over a total of six steps. As before, the cells being processed in the current step are shown as black and require the previously calculated grey cells. The processing order of the cells is dependent on the direction of the sweep, and is not the same for sweeps that originate in the same octant (as was the case for structured meshes). The incoming data to a cell are determined by the mesh geometry, and it is apparent that the propagation speed of the wavefronts also varies with direction. The same situation occurs with three-dimensional geometries, only with the mesh being composed of tetrahedrons, pyramids, hexahedrals and prisms.

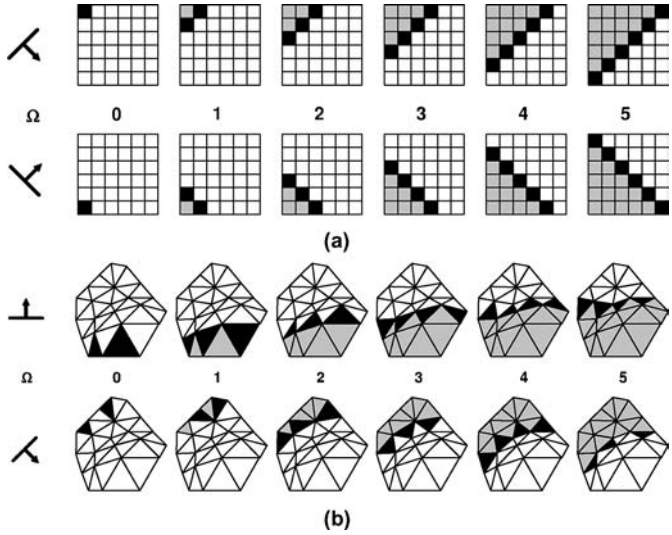

Figure 1. Example sweep processing in two dimensions for structured (a) and unstructured (b) meshes.

2.2. *Sweeping in parallel*

Parallel wavefront computations exhibit a balance between processor efficiency and communication cost. Faster wavefronts, generated by a data decomposition leading to small subgrid sizes per processor, introduce higher communication costs but result in high processor utilization. The opposite holds true for slower moving sweeps due to larger subgrid sizes. In order to optimize wavefront dynamics, S_N transport applications typically utilize blocking of the spatial subgrid, the wavefront angle set, and/or energy group set (if applicable).

Structured meshes. In codes such as Sweep3D which perform an S_N transport computation on a structured 3-D mesh, the mesh is mapped onto a logical 2-D processor array such that each processor has a column of data which can be blocked to improve parallel efficiency. The processing is effectively synchronized after the first sweep has moved across the processor array resulting in all processors being active. The processing involved in each sweep is dependent on the block size (a known constant). Thus one diagonal of the logical 2-D processor array will be processing one sweep (or one block in the third mesh dimension) while the previous diagonal is processing the next sweep and so on as shown in Figure 2(a). The direction of sweep travel is again indicated by Ω and inter-processor communications shown by arrows.

Unstructured meshes. The processing on an unstructured mesh can follow the same dependency rules as above, but the mesh partitioning is typically done in all three dimensions. An example 2-D unstructured mesh partitioned in both dimensions is shown in Figure 2 (b).

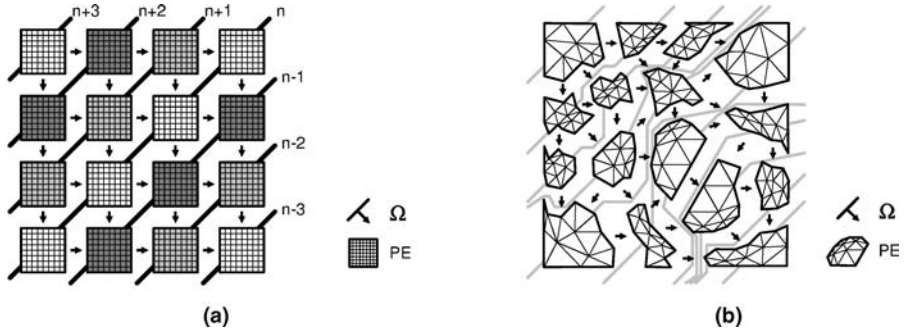

Figure 2. The pipeline effect of processing sweeps in parallel for structured (a) and unstructured (b) meshes.

The communication between processors is shown by arrows, and a simplified propagation of the sweep in the indicated direction is shown by the grey lines. A sweep for each direction (angle) is required. The unit of computational work can be considered as a single cell-angle pair.

The processing on an unstructured mesh can also be blocked—up to a set maximum number of cell-angle pairs can be processed per step. This blocking is analogous to the blocking in the third dimension in the case of the structured mesh. However, it can result in processor inefficiency down the pipeline—there is no guarantee that the same number of cell-angle pairs will be processed by all processors in a step resulting in possible processor idleness.

3. Processing characteristics of S_N transport

The main characteristics of S_N transport on structured and unstructured meshes are:

mesh partitioning The mesh partitioning strategy will determine the number and location of processor neighbors and hence communication costs for boundary data transfer. The partitioning may also cause load imbalance for unstructured grids since the number of cells that can be processed in each step on each PE may be different.

pipeline processing A processor is inactive until the edge of a sweep enters cells in that processor's partition. However, multiple sweeps may be active at any given time in the processor array. Overlap exists between computation and communication within each sweep, and across the active sweeps.

processor utilization The processing dependency in the direction of each sweep may lead to downstream PEs waiting for the upstream PEs to send the necessary boundary information. This situation can lead to processors being starved of work waiting for the results from other PEs.

strong scaling S_N transport calculations may be executed in a strong scaling mode. This causes a change in the actual use of the memory hierarchy since the number of cells per processor will decrease while increasing the number of processors used.

An understanding of these factors is required in order to formulate a performance model. There are differences between structured and unstructured meshes as well as between implementations (e.g., the Tycho and UMT2K implementations considered in this work).

3.1. Mesh partitioning

The common approach in the structured case is to use a logical 2-D processor array, thus partitioning a 3-D mesh into "columns". While counter to the traditional domain decomposition approach (which seeks to minimize boundary area), this strategy works well due to the directional dependences imposed by the sweep. The communication pattern induced by this partitioning strategy can be seen in Figure 3(c).

In the unstructured case the partitioning is not done directly by the application, rather the Metis partitioner [4] is used. This mesh partitioner aims to produce equally sized partitions (equal number of cells) while minimizing boundaries. In general such an optimal partitioning of the mesh keeps the work across processors constant and minimizes the communication cost. However, due to the pipeline processing and load-balancing characteristics

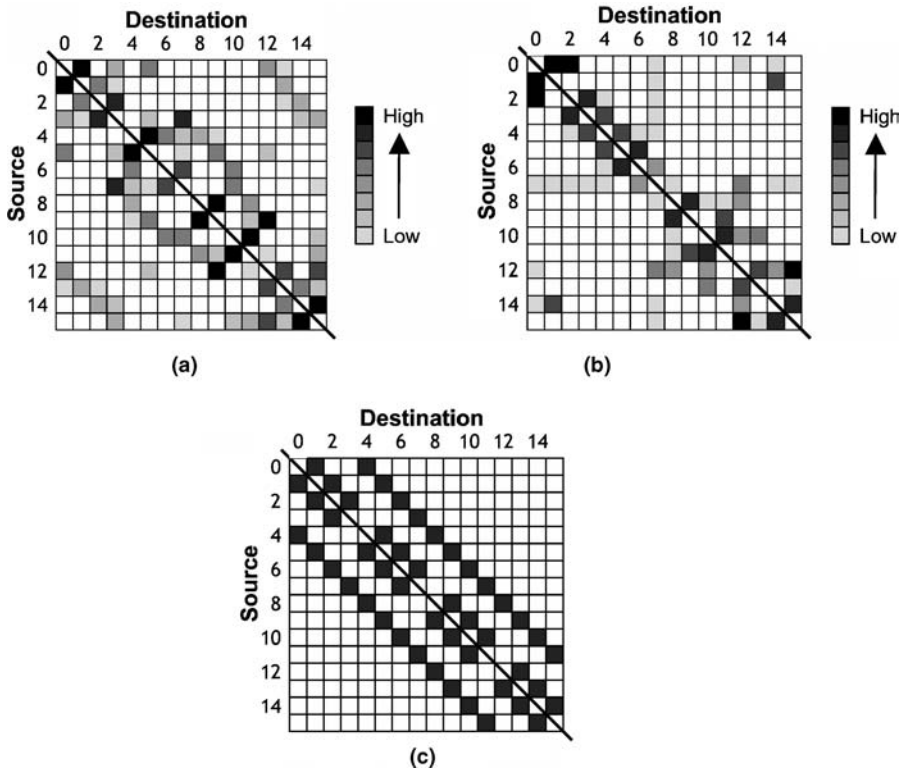

Figure 3. Example communication traffic for UMT2K (a), Tycho (b) and Sweep3D (c).

in Tycho and UMT2K as well as S_N transport in general, this partitioning may not be optimal.

For a 3-D partitioning produced by Metis, the number of cells per partition can be taken to be $E_p = N/P$ where N is the total number of cells in the mesh, and P is the number of PEs (which is equal to the number of partitions). Each 3-D partition would ideally have six nearest neighbors (equal to that of a 3-D partitioned structured mesh) each with a boundary size of $E_p^{2/3}$ cells. This essentially approximates the unstructured mesh to a structured one.

The manner in which the mesh is partitioned across processors will determine the resulting communication pattern of the application. The communication matrices in Figure 3 illustrate the effect of this approach for a 16 processor example. The existence of communication between processors indicates that they share a boundary and must therefore communicate with each other. Note that adjacent cells in the mesh are not necessarily located on logically adjacent processors. Since both unstructured mesh applications use Metis, it is not surprising that the two applications have similar, though not identical communication matrices. The shading in Figure 3 is used to illustrate the relative number of messages between communicating processors.

However, the number of communications per iteration will depend on the application. For example, Tycho implements true sweeps, communicating boundary data as needed (generating up to 64 messages per iteration in this case). UMT2K, on the other hand, only communicates once per iteration. This requires UMT2K to use old boundary information in some cases (if updated values are not yet available). This compromise can increase communication efficiency (by sending fewer messages) at the cost of potentially increasing the time to convergence. Likewise, the size of the messages will vary. In both cases, the maximum message size is approximately the same (about 3200 words). However, a larger percentage of the messages sent in UMT2K tend to be large messages while the reverse is true for Tycho.

3.2. Pipeline processing

The first cell-angle pairs processed in Tycho are those that lie on the boundary of the overall spatial mesh—those cells that have no inflows in the direction of the sweep. This corresponds to nearly all boundary elements. The sweeps thus generally start from the surface of the mesh and work their way to the center before propagating out to the opposite side of the mesh.

The dynamics of the pipeline is determined by the pipeline length and by the amount of computation done on each mesh partition. The pipeline length is determined by the number of stages in the propagation of the sweep from one side of the mesh to another. In the 2-D example shown in Figure 2(b) the number of grey lines represent the number of stages. In general, given an ideal 3-D mesh partitioning, the pipeline length is given by:

$$P_L = (P_x - 1) + (P_y - 1) + (P_z - 1) \tag{1}$$

where P_x, P_y and P_z are the number of PEs in each of the three dimensions respectively and $P = P_x * P_y * P_z$. The total work done, or the total number of cell-angle pairs processed,

in each mesh partition in an iteration of the S_N transport computation is equal to:

$$W_p = E_p * N_\Omega$$

where N_Ω is number of sweep directions. For a specific mesh, the pipeline length, P'_L can be obtained by inspection of the mesh after the partitioning has been performed. P'_L is equal to the maximum number of PEs traversed in any sweep direction and will in general be greater than P_L. The total amount of work done per partition remains as above. Both Tycho and UMT2K allow sweeps in multiple directions to be processed in parallel (using a multithreaded sweep kernel).

It is worth noting some characteristics of the pipeline length with respect to structured meshes. Equation (1) represents a lower bound on the pipeline length. It is also precise for a structured or regular input mesh. This is the special case previously examined in [2]. The common approach in the structured case is to use a logical 2-D processor array, thus partitioning the mesh into columns and decreasing the basic pipeline length to $P_x + P_y - 1$. This still assumes that all sweeps start simultaneously which is usually not the case. Pairs of directions originating in the same column are processed together (pipelined) and the consecutive directions are processed sequentially.

In actual fact, consecutive directions are overlapped such that subsequent directions may start as soon as possible, without waiting for previous directions to completely finish. This somewhat complicates the expression for pipeline length. Pairs of directions sharing a column are processed together so the delay is only 1. Consecutive 'up' and 'down' directions require $P_y - 1$ steps. Likewise, the 'down' to 'up' transitions take $P_x + P_y - 2$ steps. So, the actual pipeline length in the structured mesh case is:

$$P_L = 2P_x + 4P_y - 6 \tag{2}$$

To the pipeline length the number of steps taken in the steady state (N_{sweep}) must be added. This is the product of the number of octants, angle blocks, and k-plane blocks:

$$N_{\text{sweep}} = 8 * \frac{N_\Omega}{a} * \frac{K}{K_z}$$

where a is the angle blocking factor, K is the size of the grid in the Z dimension, and K_z is the K-plane blocking factor. Eight is the number of distinct ordinates. By substitution, the total number of steps for the structured grid case is:

$$N_{\text{steps}} = N_{\text{sweep}} + 2P_x + 4P_y - 6 \tag{3}$$

3.3. Processor utilization

Each step in both Tycho and UMT2K consists of three stages: process the cell-angle pairs which have available their in-flow boundary data, send boundary data that is produced, and

receive boundary data from neighbor sub-grids. In the performance model we assume that these stages are separated by barriers (see Figure 5). As we have shown in Section 3.1 this is an accurate representation of what happens in UMT2K, but only approximates the processing flow in Tycho.

The processing situation is complicated by the processing dependence between upstream and downstream cells in the sweep directions. This dependence may lead to downstream PEs waiting for the upstream PEs to send the necessary boundary information. In general there will be a degree of inefficiency in this operation and processors will be starved of work waiting for the results from other PEs. It is interesting to note that for the case of structured meshes, the work on each PE is equal throughout and thus the processors are fully utilized once the pipeline is filled. To quantify this inefficiency, the Parallel Computational Efficiency, *PCE* [13], is used:

$$PCE = \frac{W_p}{\sum_{S=1}^{N_{\text{steps}}} \max_P(\|work(P, S)\|)}$$

where $work(P, S)$ is the number of cell-angle pairs processed on processor P in step S, and W_p is the total number of cell-angle pairs processed on each processor in an iteration. *PCE* represents the fraction of the maximum number of cells that are processed in all steps in an iteration. When $PCE = 1$, the parallel efficiency is 100%—this can only occur on a small processor run (typically < 9 PEs) or on structured meshes. The lower the value of *PCE*, the greater the inefficiency.

A value for *PCE* can be obtained for a specific mesh after its partitioning and before the S_N transport calculation. The number of work steps required to perform the total number of cell-angle pairs per PE is given by:

$$N_{\text{sweep}} = \frac{W_p}{(MCPS * PCE)}$$

where *MCPS* is the maximum number of cells that can be processed in a step. In the general case, i.e. without pre-inspection of the mesh, a value for *PCE* has to be assumed—possibly based on experience from prior meshes. This assumption can be inaccurate reflecting the tradeoff between generality and accuracy that is present in performance modeling work.

Figure 4 illustrates the number of cells processed on each of 16 processors, up to *MCPS* in each step of the computation. The height of each column indicates the actual number of cells processed on each processor in each step. In general, the boundary processors (e.g., PEs 6–9) will be able to process close to *MCPS* cells at the beginning and end of the iteration. Likewise, interior processors (all others) will be able to process close to *MCPS* cells in the middle of the iteration once the sweep wavefronts have reached them.

3.4. Strong scaling

S_N transport calculations on unstructured meshes may be executed in a strong scaling mode i.e. parallelism is used to solve the same problem but in reduced time. The input mesh geometry does not change and thus partitions become smaller on larger processor counts.

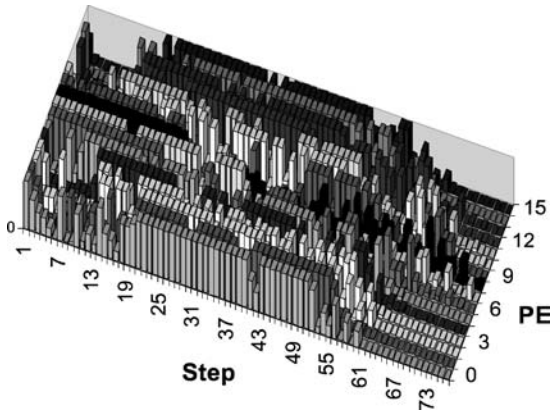

Figure 4. Parallel computational efficiency.

Strong scaling causes a change in the actual use of the memory hierarchy when increasing the number of processors used and hence its performance has to be carefully considered. For instance when a mesh partition becomes small enough to fit in cache the performance will be better than if main memory has to be accessed.

Figure 6(a) shows the computation time per cell for a number of different meshes and partition sizes for Tycho an 833 MHz Alpha EV68 processor and Figures 8(a) and 7(a) show the performance of UMT2k on a 1 GHz Itanium-2 processor and an 833 MHz Alpha EV68 processor respectively. The Alpha EV68 has an 8 MB L2 cache and the Itanium-2 used has a 3 MB L3 cache. In each case there are three regions evident: when the partition does not fit into cache (right hand plateau of the curves), when the mesh fits into cache (left hand plateau), and when partial cache re-use occurs (middle region). Note that there is also a further parameter in the case of UMT2K which is the number of energy groups that are processed per cell. This can vary and increases the processing time per cell by a multiplicative factor as shown in Figures 8(a) and 7(a). The number of energy groups in Tycho is fixed at one. Table 1 lists the analytical form of the time per cell for the model curves shown in Figures 6(a), 7(a), and 8(a).

Table 1. Time per cell for each validation configuration

Tycho/AlphaServer ES40	$T_e(E_p, G)(\mu s) =$	$G * 9.2$	$E_p \geq 16\ K$
		$G * (1.8Ln(E_p) - 8.4)$	$800 < E_P < 16\ K$
		$G * 3.7$	$E_P \leq 800$
UMT2K/Itanium-2 Cluster	$T_e(E_p, G)(\mu s) =$	$(3 + G) * 0.14$	$E_p \geq 50\ K$
		$(3 + G) * (0.02Ln(E_p) - 0.09)$	$12\ K < E_P < 50\ K$
		$(3 + G) * 0.11$	$E_P \leq 12\ K$
UMT2K/AlphaServer ES40	$T_e(E_p, G)(\mu s) =$	$(1 + G) * 0.0224$	$E_p \geq 340\ K$
		$(1 + G) * (0.0028Ln(E_p) - 0.0136)$	$10\ K < E_P < 340\ K$
		$(1 + G) * 0.012$	$E_P \leq 10\ K$

There is some variation in performance when using different meshes in this analysis due to the different memory access patterns and hence the actual cache reuse. It can be seen that a good approximation to the the time taken to process a cell can be encapsulated in a piece-wise linear model. In general however, we are interested in large meshes - those that unfortunately will not exhibit cache re-use, and also those that cannot be executed on small processor counts due to limitations in the size of the actual memory per processor.

4. Analytical performance model

In the performance model of S_N transport computations we assume that the three stages that constitute a processing step are distinct and do not overlap—those of computation, blocking sends and blocking receives for the boundary communications. Further the model also takes the maximum of the amount of work performed in each step over all processors, as well as the maximum of the communications performed to/from any one node. This scenario is illustrated by Figure 5. This assumption will tend to give an over prediction of the iteration time. The model for the time of a single iteration is formulated as:

$$T_i = \left(\sum_{S=1}^{N_{\text{steps}}} \max_P(Work(P, S)) \right) . T_e\left(\frac{N}{P}, G\right) + \sum_{S=1}^{N_{\text{steps}}} \max_P \left(\sum_{C=1}^{\|N_c(S,P)\|} T_c(N_c(S, P, C), N_s(S, P, C)) \right) \tag{4}$$

where the first term represents computation and the second term communication. There are N_{steps} steps per iteration with $Work(P, S)$ cell-angle pairs being processed on processor P in step S. The time to process x cell-angle pairs for G energy groups is given by $T_e(x, G)$. $N_c(S, P, C)$ is the destination PE for communication C in step S on processor P and $N_s(S, P, C)$ is the size of the same communication. $\|N_c(S, P)\|$ is the total number of communications originating from processor P in step S. The time to communicate a message of size y bytes to processor x is given by $T_c(x, y)$.

The differences between the two S_N transport implementations become apparent in the way in which the parameters to this model are specified. The number of steps in general is

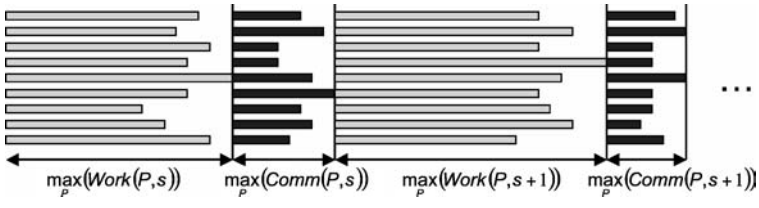

Figure 5. Load imbalance.

given by:

$$N_{\text{steps}} = N_{\text{sweep}} + P_L$$

For Tycho this is:

$$N_{\text{steps}} = \left(\frac{W_P}{MCPS * PCE}\right) + (P_x - 1) + (P_y - 1) + (P_z - 1)$$

where the first part of the equation represents the number of steps required to processes a sub-grid given a value of PCE, and the second part is the length of the pipeline in 3-D. The number of steps in UMT2K is given by:

$$N_{\text{steps}} = N_\Omega * noutmx$$

where N_Ω is the size of the discrete ordinates set, and $noutmx$ is the number of outer iterations. This is independent of the pipeline length since UMT2K does not implement a strict sweep operation—it uses old boundary data from the previous time-step which also has the effect of making $PCE = 1$ if the partitioner produces equal sized partitions. It can be considered as a simpler S_N transport implementation than Tycho. It does not require the same complexity in the handling of the boundary data flow which is reflected in the difference in the time taken to process a single element shown in Figures 6(a) and 7(a).

The communications per step in both Tycho and UMT2K are approximated to be similar to that of a partitioned structured 3-D mesh and remain constant throughout. This will in general under predict the actual communication time. A structured grid has 6 local neighbors resulting from a 3-D partitioning of a 3-D spatial grid. The size of the boundary on each boundary surface is $E_p^{2/3}$ cells.

The parameters of $T_e()$, and $T_{\text{comm}}()$ are specific to a particular system and are measured. A two-parameter, piece-wise linear model for the communication is assumed which uses the Latency (L_c) and Bandwidth (B_c) of the network communication.

$$T_{\text{comm}}(S) = L_c(S, D) + S.\frac{1}{B_c(S, D)}$$

where L_c is the communication latency, B_c is the communication bandwidth, and S is the message size.

The communication time is subject to contention in the communication network. Our experience using UMT2K and Tycho as well as other codes on clusters of SMPs, is that the main contention occurs on the number of out-of-node communications that occur simultaneously. For example with the fat-tree interconnect of the Quadrics network [14], the number of communications that collide in higher levels of the fat-tree is low due to dynamic routing. The contention is taken into account by a multiplicative constant on the communication time, T_{comm}, which represents the number of simultaneous out-of-node communications.

4.1. Performance model validation

The performance model described in Section 4 is validated on a 64 node AlphaServer ES40 cluster for both Tycho and UMT2K. We further cross-validate the performance of UMT2K

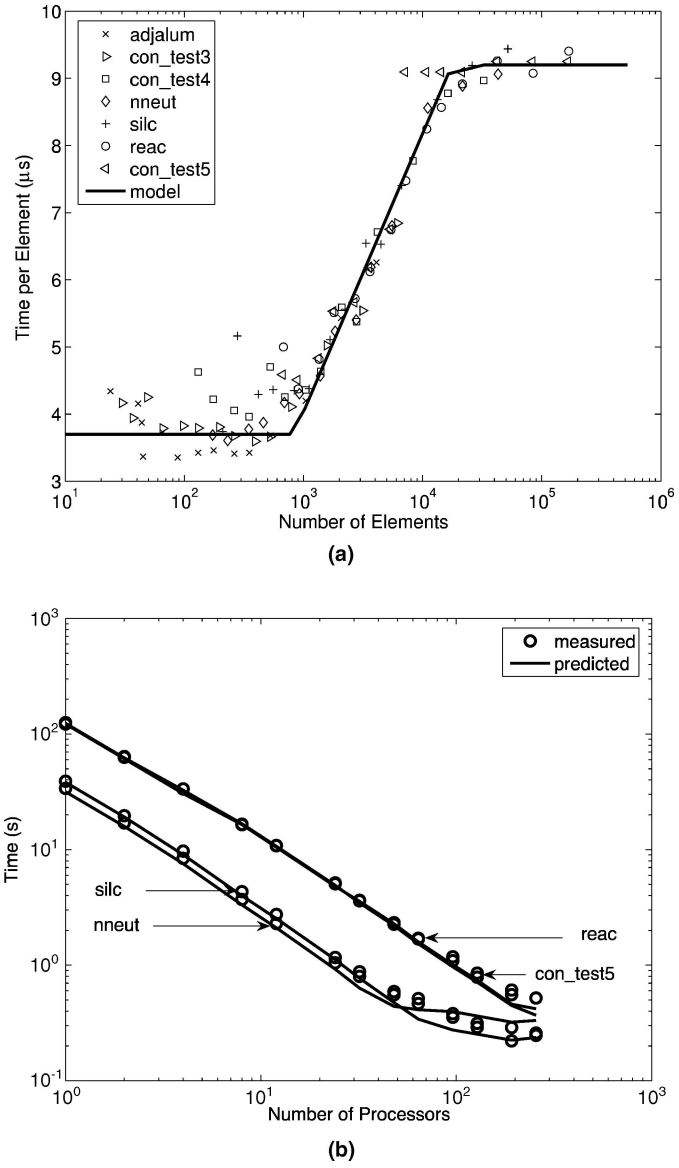

Figure 6. Time per cell (a) and model validations (b) for Tycho on the AlphaServer ES40.

on a 32 node Itanium-2 cluster. The Itanium-2 cluster consists of 2 processors per node running at 1 GHz each with a 256 K L2 cache, a 3 MB L2 cache, and 2 GB main memory. The AlphaServer cluster consists of 4 processors per node running at 833 MHz each with an 8 MB L2 cache and 2 GB main memory. The nodes in both the clusters are interconnected using the Quadrics QSnet-I high speed network with Elan3 switching technology. The performance characteristics of these systems are listed in Table 2.

Several unstructured meshes are used in the validation process. For UMT2K, the input meshes consist of a 2-D mesh of triangles which are projected into the third dimension

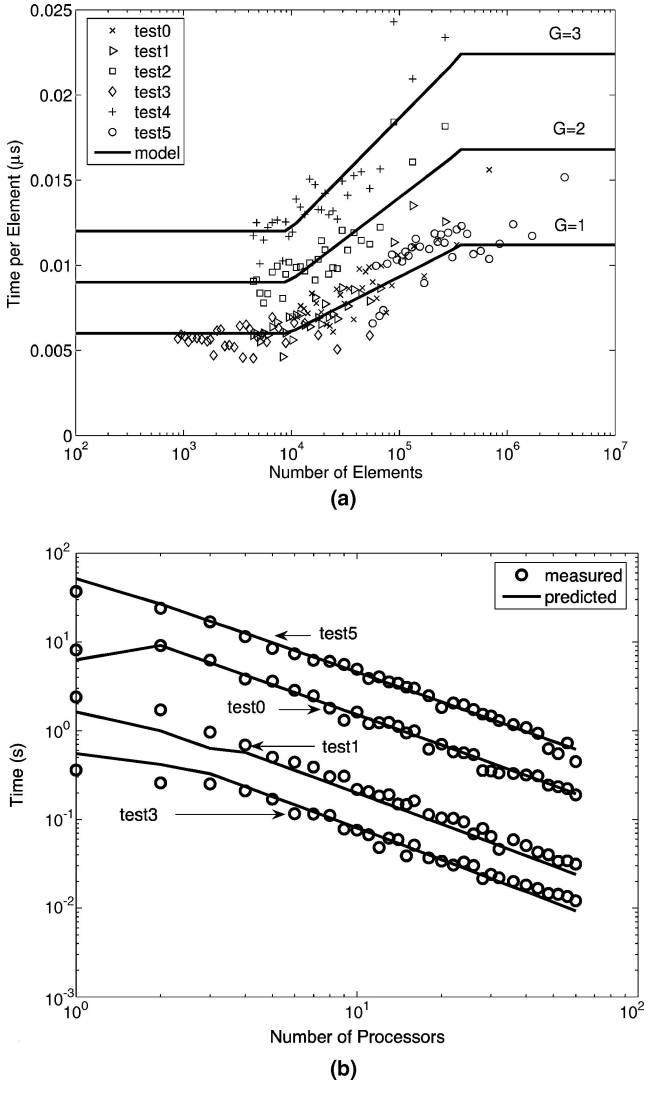

Figure 7. Time per cell (a) and model validations (b) for UMT2K on the AlphaServer ES40.

to form multiple layers of cells. For Tycho, the input meshes consist of a 3-D mesh of tetrahedrals. Six mesh configurations are considered for UMT2K which arise fom two different meshes which are projected by different amounts in the third dimension, and also differ in the number of energy groups processed per cell. Four different meshes are considered for Tycho. The tests are chosen to represent a range of partition sizes and are listed in Tables 3 and 5.

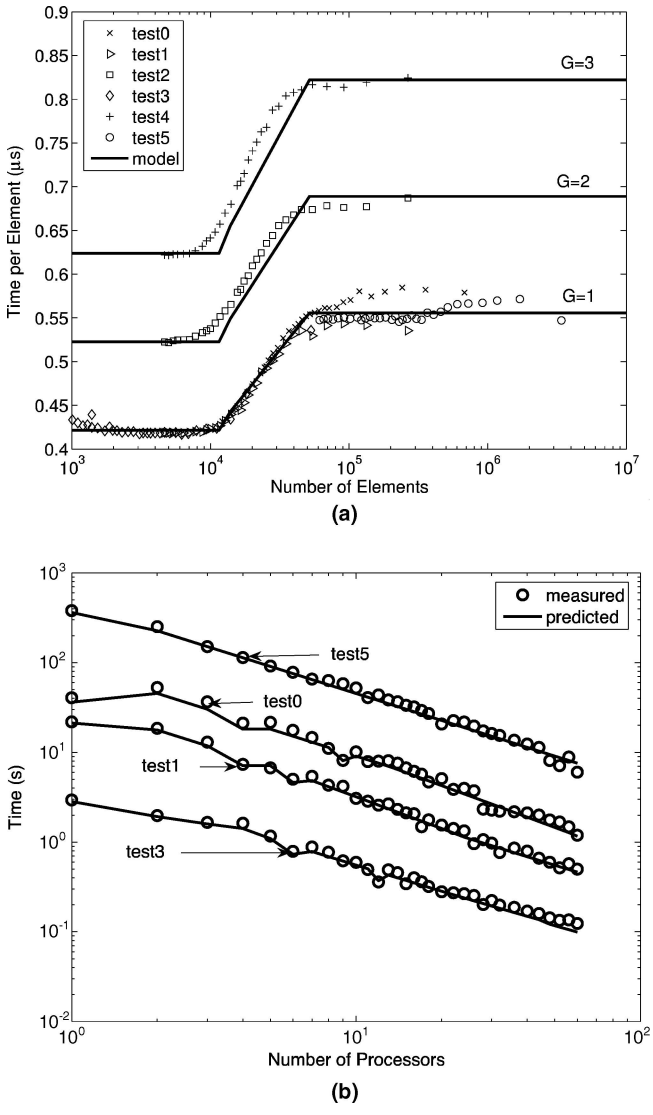

Figure 8. Time per cell (a) and model validations (b) for UMT2K on the Itanium-2 cluster.

Table 2. Hardware parameters for both validation systems

	Itanium-2 1GHz	AlphaServer ES40 833 MHz
$L_c(S)(\mu s)$	$\begin{cases} 6.48 & S < 64B \\ 8.21 & 64 \leq S \leq 256 \\ 17.1 & S > 512 \end{cases}$	$\begin{cases} 9.28 & S < 64B \\ 9.00 & 64 \leq S \leq 256 \\ 21.4 & S > 512 \end{cases}$
$1/B_c(S, D)(ns)$	$\begin{cases} 0.0 & S < 64B \\ 25.5 & 64 \leq S \leq 512 \\ 13.7 & S > 512 \end{cases}$	$\begin{cases} 0.0 & S < 64B \\ 22.7 & 64 \leq S \leq 512 \\ 11.2 & S > 512 \end{cases}$

Table 3. Test cases for the validation of the Tycho performance model (alpha)

Case	Mesh	#Cells	Description	Error (%)
6	Nneut	43,012	Neutron well-logging tool and surrounding media	13.48
7	Silc	51,963	Computer Chip and packaging for radiation shielding	12.07
8	Reac	165,530	Reactor pressure vessel and surrounding cavity structures	7.44
9	Con_test5	168,356	Cube divided into approximately equal-sized elements	8.07

Table 4. Test cases for the validation of the UMT2K performance model (alpha)

Case	Mesh	#Cells	Description	Error (%)
0	MMesh	680,400	Medium mesh, 4950 cells/layer, 3 layers, 1 energy group	9.74
1	SMesh	265,680	Small mesh, 398 cells/layer, 15 layers, 1 energy group	10.90
2	SMesh	265,680	Small mesh, 398 cells/layer, 15 layers, 2 energy groups	9.49
3	SMesh	53,136	Small mesh, 398 cells/layer, 3 layers, 1 energy group	16.94
4	SMesh	265,680	Small mesh, 398 cells/layer, 15 layers, 3 energy groups	8.86
5	MMesh	3,402,000	Large mesh, 4950 cells/layer, 15 layers, 1 energy group	11.31

Measurements and model predictions for the four test cases on Tycho are shown in Figure 6(b) and for four of the test cases on UMT2K are shown in Figure 7(b). The cross-hardware validation of UMT2K on the Itanium-2 cluster is shown in Figure 8(b). A summary of the difference between the model predictions and the measurements is listed in the last column of Tables 3, 4, and 5. It can be seen that the model predicts each case with high accuracy with an average error of approximately 11% across all the test cases.

4.2. Structured meshes

Given Equation (4) it is possible to derive a model for the structured mesh case. First, the max functions are not required since each processor is assigned an equal number of cells and performs the same amount of work per step. Likewise, each processor has the same number of neighbors and therefore the same amount of communication. This reduces Equation (4)

Table 5. Test cases for the validation of the UMT2K performance model (ia64)

Case	Mesh	#Cells	Description	Error (%)
0	MMesh	680,400	Medium mesh, 4950 cells/layer, 3 layers, 1 energy group	12.53
1	SMesh	265,680	Small mesh, 398 cells/layer, 15 layers, 1 energy group	8.33
2	SMesh	265,680	Small mesh, 398 cells/layer, 15 layers, 2 energy groups	8.98
3	SMesh	53,136	Small mesh, 398 cells/layer, 3 layers, 1 energy group	11.41
4	SMesh	265,680	Small mesh, 398 cells/layer, 15 layers, 3 energy groups	8.87
5	MMesh	3,402,000	Large mesh, 4950 cells/layer, 15 layers, 1 energy group	9.06

to:

$$T_i = \left(\sum_{S=1}^{N_{\text{steps}}} \frac{I}{P_x} \frac{J}{P_y} \frac{K}{K_z} \frac{N_\Omega}{a} \right) * T_e\left(\frac{N}{P}, G \right) + \left(\sum_{S=1}^{N_{\text{steps}}} 4 * T_c(N_c(S, P, C), N_s(S, P, C)) \right)$$

where I, J, and K are the three dimensions of the structured grid, P_x and P_y are the dimensions of the 2-D processor grid, K_z is the plane blocking factor, N_Ω is the total number of angles and a is the angle blocking factor. We can simplify this by substituting

$$T_{\text{cpu}} = \frac{I}{P_x} \frac{J}{P_y} \frac{K}{K_z} \frac{N_\Omega}{a} * T_e\left(\frac{N}{P}, G \right)$$

and

$$T_{\text{msg}} = T_c(N_c(S, P, C), N_s(S, P, C))$$

giving

$$T_i = N_{\text{steps}} * (T_{\text{cpu}} + 4 * T_{\text{msg}})$$

substituting for N_{steps} from Equation (3) results with

$$T_i = (N_{\text{sweep}} + 2 * P_x + 4 * P_y - 6) * (T_{\text{cpu}} + 4 * T_{\text{msg}}) \tag{5}$$

One further refinement is required. Recall that boundary processors will each have two neighbors while interior processors will each have four. As formulated, we assume that every processor has four neighbors. To remedy this, we must separate Equation (5) into two terms, one for pipeline fill steps and one for the steady state. Note that the pipeline length is limited by the perimeter of the processor grid. It follows that these steps will involve boundary processors with only two neighbors. This gives a final formulation of:

$$T_i = N_{\text{sweep}} * (T_{\text{cpu}} + 4 * T_{\text{msg}}) + (2 * P_x + 4 * P_y - 6) * (T_{\text{cpu}} + 2 * T_{\text{msg}})$$

5. Summary

In this work we have presented a predictive analytical performance and scalability model for S_N transport computations on unstructured meshes. These calculations are representative of a high proportion of cycles used across all ASC systems. Thus modeling and understanding their performance is important not only on current systems, but also looking ahead to possible larger scale systems in the future.

The performance model has been shown to be accurate with a typical error of 11% across a range of configurations in terms of processor count, mesh geometry, and systems utilized. Further, the model is shown to be applicable to two quite different implementations of these types of computations. The implementations differ in their input meshes, and in the actual type of S_N transport calculation performed. The differences are handled by changing the parameter inputs to the analytical performance model. It has also been shown that the model also represents the performance of the simplier case of structured meshes.

We believe performance modeling is key to building performance engineered applications and architectures. This work is one of few performance models that exist for entire applications. It follows on from our work on structured particle transport modeling [2], adaptive mesh refinement modeling [5], and Monte-Carlo simulation [12].

Acknowledgments

We would like to thank Shawn Pautz for many informative discussions on the implementation details of Tycho. This work was supported in part by a Los Alamos LDRD 2001609DR "Performance Analysis and Modeling of Extreme-Scale Parallel Architectures" and by the Defense Advanced Research Projects Agency under the High Productivity Computing Systems Program. Mark Mathis is currently a PhD candidate at Texas A&M University and is supported in part by a Department of Energy High Performance Computer Science Fellowship. Los Alamos National Laboratory is operated by the University of California for the U.S. Department of Energy under contract W-7405-ENG-36.

References

1. K. Davis, A. Hoisie, G. Johnson, D. J. Kerbyson, M. Lang, S. Pakin, and F. Petrini. A performance and scalability analysis of the BlueGene/L architecture. In *Proc. IEEE/ACM Supercomputing*, Pittsburgh, PA, 2004.
2. A. Hoisie, O. Lubeck, and H. Wasserman. Performance and scalability analysis of Teraflop-scale parallel architectures using multidimensional wavefront applications. *Int. J. of High Performance Computing Applications*, 14(4):330–346, 2000.
3. A. Hoisie, O. Lubeck, H. Wasserman, F. Petrini, and H. Alme. A general predictive performance model for wavefront algorithms on clusters of SMPs. In *Proc. of ICPP 2000*, pages 20–25, Toronto, Canada, 2000.
4. G. Karypis and V. Kumar. METIS 4.0: Unstructured Graph Partitioning and Sparse Matrix Ordering System. Technical report, Department of Computer Science, University of Minnesota, 1998.
5. D. J. Kerbyson, H. Alme, A. Hoisie, F. Petrini, H. Wasserman, and M. Gittings. Predictive performance and scalability modeling of a large-scale application. In *Proc. Supercomputing*, Denver, CO, 2001.
6. D. J. Kerbyson, A. Hoisie, and H. J. Wasserman. Modeling the performance of large-scale systems. *IEE Proceedings (Software)*, 150(4):214–221, 2003.

7. D. J. Kerbyson, A. Hoisie, and H. J. Wasserman. A performance comparison between the earth simulator and other terascale systems on a characteristic ASCI workload. *Concurrency and Computation, Practice and Experience*, 17(10):1219–1238, 2004.
8. D. J. Kerbyson, A. Hoisie, and H. J. Wasserman. Use of predictive performance modeling during large-scale system installation. *To appear in Parallel Processing Letters*, 2005.
9. K. R. Koch, R. S. Baker, and R. E. Alcouffe. Solution of the first-order form of the 3D discrete ordinates equation on a massively parallel processor. *Transactions of the American Nuclear Society*, 65:198–199, 1992.
10. M. M. Mathis, N. M. Amato, and M. L. Adams. A general performance model for parallel sweeps on orthogonal grids for particle transport calculations. In *Proc. ACM Int. Conf. Supercomputing (ICS)*, pp. 255–263, Santa Fe, NM, 2000.
11. M. M. Mathis and D. J. Kerbyson. Performance modeling of unstructured mesh particle transport computations. In *Proc. ACM/IEEE Int. Parallel and Distributed Processing Symposium (IPDPS)*, Santa Fe, NM, 2004.
12. M. M. Mathis, D. J. Kerbyson, and A. Hoisie. A performance model of non-deterministic particle transport on large-scale systems. In *Proc. Int. Conf. on Computational Science (ICCS)*, LNCS, vol. 2659, pp. 936–945, Melbourne, Australia, 2003.
13. S. D. Pautz. An algorithm for parallel sn sweeps on unstructured meshes. *J. Nuclear Science and Engineering*, 140:111–136, 2002.
14. F. Petrini, W. C. Feng, A. Hoisie, S. Coll, and E. Frachtenberg. The Quadrics Network: High-Performance Clustering Technology. *IEEE Micro*, 22(1):46–57, 2002.
15. F. Petrini, D. J. Kerbyson, and S. Pakin. The case of the missing supercomputer performance: Achieving optimal performance on the 8,192 processors of ASCI Q. In *Proc. IEEE/ACM SuperComputing*, Phoenix, 2003.
16. S. Plimpton, B. Hendrickson, S. Burns, and W. McLendon. Parallel algorithms for radiation transport on unstructured grids. In *Proc. IEEE/ACM Supercomputing*, Dallas, 2000.
17. *The ASCI SWEEP3D README File*. Available from: www.llnl.gov/asci_benchmarks/asci/limited /sweep3d/sweep3d_readme.html
18. *The UMT2K (UMT 1.2) README File*. Available from: www.llnl.gov/asci/purple/benchmarks/ limited/umt/ umt1.2.readme.html
19. J. S. Vetter and A. Yoo. An empirical performance evaluation of scalable scientific applications. In *Proc. IEEE/ACM Supercomputing*, Baltimore, MD, 2002.